PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-006043

(43)Date of publication of application : 10.01.1995

(51)Int.Cl. G06F 9/46
G06F 13/00

(21)Application number : 06-034742 (71)Applicant : MITSUBISHI ELECTRIC CORP

(22)Date of filing : 04.03.1994 (72)Inventor : MIZUNUMA ICHIRO
SHIMAKAWA HIROMITSU

(30)Priority

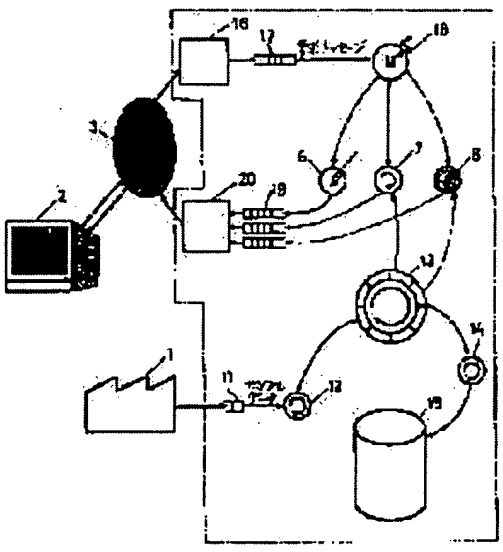
Priority number : 05 69055 Priority date : 05.03.1993 Priority country : JP

(54) MULTITHREAD SERVER

(57)Abstract:

PURPOSE: To perform processing s for request messages even while received sample data are stored in a memory, and to speed up the processing s and prevent a request message from being absent by taking out the request messages, held by a 2nd reception part, in order and performing the processing s corresponding to their contents on the basis of the sample data which are received by a 1st reception part and stored in the memory.

CONSTITUTION: The request messages are sent at irregular intervals from a client 2 through



a network 3 and received by a receiver 16. Then the request messages received by the receiver 16 are temporarily stored in an FIFO queue 17. A mother thread 18 takes in the stored request messages in order and generates and actuates one of child threads 6, 7, and 8 according to the contents. The actuated child thread 6, 7, or 8 performs the processing corresponding to the contents of the request messages on the basis of the sample data stored in a ring buffer 13 and on a disk 15.

LEGAL STATUS

[Date of request for examination]	04.03.1994
[Date of sending the examiner's decision of rejection]	24.06.1997
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平7－6043

(43)公開日　平成７年(1995)１月10日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/46	３４０　Ｂ	8120－5B		
	13/00	３５７　Ｚ	7368－5B		

審査請求　有　　請求項の数10　ＯＬ　（全 24 頁）

(21)出願番号　特願平6－34742

(22)出願日　平成６年(1994)３月４日

(31)優先権主張番号　特願平5－69055

(32)優先日　平５(1993)３月５日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

(72)発明者　水沼　一郎
尼崎市塚口本町８丁目１番１号　三菱電機株式会社産業システム研究所内

(72)発明者　島川　博光
尼崎市塚口本町８丁目１番１号　三菱電機株式会社産業システム研究所内

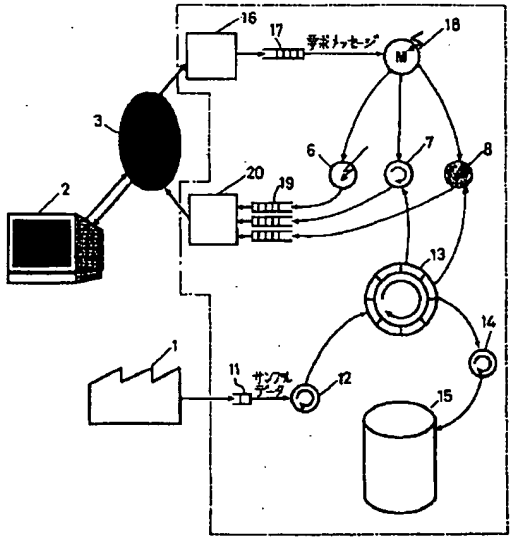
(74)代理人　弁理士　田澤　博昭　（外１名）

(54)【発明の名称】　マルチスレッド・サーバ

(57)【要約】

【目的】　受信したサンプルデータをメモリに格納中であっても、要求メッセージに対する処理を可能にして、要求メッセージに対する処理の高速化を図ることができるとともに、要求メッセージの欠損を防止できるマルチスレッド・サーバを得ることを目的とする。

【構成】　第２の受信部１６・１７により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を第１の受信部１１が受信してメモリ１３・１５に格納したサンプルデータに基づいて行うようにしたものである。



1：プラント
2：クライアント（外部装置）
3：ネットワーク（外部装置）
6,7,8：チャイルドスレッド（処理実行部）
11：FIFOキュー（第1の受信部）
13：リングバッファ（メモリ）
15：ディスク（メモリ）
16：レシーバ（第2の受信部）
17：FIFOキュー（第2の受信部）
18：マザースレッド（処理実行部）
19：FIFOキュー（送信部）
20：トランスミッタ（送信部）

【特許請求の範囲】

【請求項１】　プラント等からサンプルデータを受信する第１の受信部と、上記第１の受信部により受信されたサンプルデータを格納するメモリと、外部装置から要求メッセージを受信するとともに、その要求メッセージを保持する第２の受信部と、上記第２の受信部により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を上記メモリに格納されたサンプルデータに基づいて行う処理実行部と、上記処理実行部の処理結果を一時的に保持したのち、その処理結果を上記外部装置に対して送信する送信部とを備えたマルチスレッド・サーバ。

【請求項２】　プラント等からサンプルデータを受信する第１の受信部と、上記第１の受信部により受信されたサンプルデータを格納するメモリと、外部装置から要求メッセージを受信するとともに、その要求メッセージを保持する第２の受信部と、上記第２の受信部により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を上記メモリに格納されたサンプルデータに基づいて行う処理実行部と、上記処理実行部の処理結果を一時的に保持するとともに、複数の処理結果を保持した場合には優先度の高い処理結果から順次上記外部装置に対して送信する送信部とを備えたマルチスレッド・サーバ。

【請求項３】　上記送信部は、上記処理実行部の処理結果を上記外部装置に対して送信する処理を行う際に、他の処理と競合する場合、該他の処理より優先度が高いときのみ処理を行うことを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項４】　上記処理実行部は、上記第２の受信部が複数の要求メッセージを保持している場合、優先度の高い要求メッセージから順次取り出すことを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項５】　上記処理実行部は、各スレッドが処理を実行する際に必要となるメモリ領域を各スレッド毎に予め確保しておき、各スレッドに処理を実行させる際に対応するメモリ領域を割り当てることを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項６】　上記処理実行部は、処理に必要となるスレッドを予め生成し、待機状態にしておくことを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項７】　上記メモリは、上記第１の受信部により受信されたサンプルデータを格納中に、上記処理実行部から所定のアドレスに格納されているサンプルデータの読み込み要求があった場合、その読み込み要求に係るアドレスが、現在サンプルデータを格納しているアドレスと一致するときのみその読み込み要求を却下することを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項８】　上記処理実行部が生成するスレッドのち、周期的に起動される周期起動型スレッドの起動及び終了を管理するタイマスレッドを設け、そのタイマスレッドは所定の起動周期毎にその周期起動型スレッドを起動させるとともに、その周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させることを特徴とする請求項１または請求項２記載のマルチスレッド・サーバ。

【請求項９】　上記処理実行部が生成するスレッドと上記タイマスレッド間の共有資源に対する読み書きの相互排除を行うためにセマフォを設けるとともに、上記タイマスレッドより優先度の低いスレッドがそのセマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能を上記セマフォに持たせたことを特徴とする請求項８記載のマルチスレッド・サーバ。

【請求項１０】　上記処理実行部が生成するスレッドと上記タイマスレッド間の共有資源に対する読み書きの相互排除を行うために、当該共有資源を使用しているか否かを示すフラグ及びそのフラグを読み書きするためのセマフォを各スレッド毎に設けるとともに、上記タイマスレッドより優先度の低いスレッドが当該セマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能を上記セマフォに持たせたことを特徴とする請求項８記載のマルチスレッド・サーバ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、外部装置からの要求メッセージに対し、リアルタイムに応答できるマルチスレッド・サーバに関するものである。

【０００２】

【従来の技術】図２２は従来のマルチスレッド・サーバを示す構成図であり、図において、１はプラント、２はクライアント、３はネットワーク（バックプレーンバスでもよい）であり、クライアント２及びネットワーク３から外部装置が構成されている。４はプラント１からサンプルデータを受信するとともに、外部装置から要求メッセージを受信するレシーバ、５はレシーバ４により受信された要求メッセージの内容に応じてチャイルドスレッド６，７，８のいずれかを起動するとともに、レシーバ４がサンプルデータを受信した場合にはそのサンプルデータをメモリ９に格納するマザースレッド、６，７，８は外部装置から送られる要求メッセージに応じた処理をメモリ９に格納されたサンプルデータに基づいて行うチャイルドスレッドであり、チャイルドスレッド６，７，８は起動時にマザースレッド５によって生成され

る。９はサンプルデータを格納するメモリ、１０はチャイルドスレッド６，７，８の処理結果を一時的に保持するＦＩＦＯキュー、２０はＦＩＦＯキュー１０に保持された処理結果を、保持された順に取り出して外部装置へ送信するトランスミッタである。

【０００３】次に動作について説明する。まず、レシーバ４は、プラント１からサンプルデータまたは外部装置から要求メッセージを受信する。そして、マザースレッド５は、レシーバ４がプラント１からサンプルデータを受信した場合には、そのサンプルデータをメモリ９に格納し、レシーバ４が外部装置から要求メッセージを受信した場合には、その要求メッセージの内容に応じてチャイルドスレッド６，７，８のいずれかを生成して起動する。

【０００４】例えば、マザースレッド５によりチャイルドスレッド６が起動された場合には、チャイルドスレッド６は、要求メッセージに応じた処理をメモリ９に格納されたサンプルデータに基づいて行う。そして、チャイルドスレッド６の処理結果がＦＩＦＯキュー１０に送信され、一時的に保持される。そして、トランスミッタ２０はＦＩＦＯキュー１０に保持された処理結果を、保持された順に取り出して外部装置へ送信する。即ち、図２３に示すように、ＦＩＦＯキュー１０に先に保持された処理結果が、後に保持された処理結果より優先度が低い場合でも、優先度の低い先に保持された処理結果から送信される。なお、チャイルドスレッド６，７，８はそれぞれ並列的に処理を行うことができるが、メモリ９に対するアクセスは同時に行うことができない。

【０００５】

【発明が解決しようとする課題】従来のマルチスレッド・サーバは以上のように構成されているので、プラント１等からのサンプルデータも外部装置からの要求メッセージもレシーバ４が受信してマザースレッド５がその後の処理を決定しなければならず、従って、例えば、受信した要求メッセージに応答する処理を行っている途中で、サンプルデータを受信した場合、その処理が終了するまでそのサンプルデータをメモリ９に格納することができず、その結果、そのサンプルデータの格納が次のサンプルデータの到着までに終了しないと、サンプルデータの欠測が起こる事態が発生する問題点があった。

【０００６】また、レシーバ４は、複数の要求信号等を保持する機能がないので、マザースレッド５が、レシーバ４により受信された要求メッセージ等（便宜上、旧要求メッセージという）を取り込む前に、他の要求メッセージ等を外部装置から送られてしまうと、旧要求メッセージが欠損してしまう問題点があった。

【０００７】また、トランスミッタ２０は、ＦＩＦＯキュー１０に保持された順に取り出して送信するので、先に保持された処理結果より後から保持された処理結果の方が高速性を要求されていても、先に保持された処理結果を送信するまで後に保持された処理結果は待たされてしまい、外部装置の要求を十分満足することができない問題点があった。

【０００８】また、トランスミッタ２０は、送信以外の他の処理を行うスレッド（図３参照）の存在を確認せずに処理結果を送信してしまうので、トランスミッタ２０の送信処理より当該スレッドの処理の方が高速性を要求されていても、トランスミッタ２０の送信処理が終わるまで当該スレッドの処理が待たされてしまう問題点があった。

【０００９】また、マザースレッド５は、要求メッセージに応答する処理を実行する際に、チャイルドスレッド６，７，８を生成する毎にチャイルドスレッド６，７，８の処理に必要なメモリ領域を確保しなければならず、従って、徐々にメモリ空間上に不要となったメモリ領域が増えていくためガベージコレクションを行う必要が生じ、その間要求メッセージに対する応答ができなくなる問題点があった。

【００１０】また、マザースレッド５は、要求メッセージに応答する処理を実行する際に、その都度チャイルドスレッド６，７，８を生成する必要があるが、スタック領域が不足しているような場合には生成することができず、スタック領域が解放されるまで処理を待機しなければならない問題点があった。

【００１１】さらに、チャイルドスレッド６，７，８はそれぞれ並列的に処理を行うことができるが、メモリ９に対するアクセスは同時に行うことができないので、並列処理の効率が低下し、高速処理ができないなどの問題点もあった。

【００１２】請求項１の発明は上記のような問題点を解消するためになされたもので、受信したサンプルデータをメモリに格納中であっても、要求メッセージに対する処理を可能にして、要求メッセージに対する処理の高速化を図ることができるとともに、要求メッセージの欠損を防止できるマルチスレッド・サーバを得ることを目的とする。

【００１３】請求項２の発明は、上記目的に加え、高速性の要求される処理結果から順次送信できるようにして、外部装置の要求を十分満足できるマルチスレッド・サーバを得ることを目的とする。

【００１４】請求項３の発明は、処理結果を外部装置へ送信する処理よりも、高速性の要求される処理がある場合には、その高速性の要求される処理を優先して処理ができるマルチスレッド・サーバを得ることを目的とする。

【００１５】請求項４の発明は、要求メッセージの欠損を防止できるとともに、高速性の要求される要求メッセージに対する処理から行うことができるマルチスレッド・サーバを得ることを目的とする。

【００１６】請求項５の発明は、ガベージコレクション

を行う必要をなくし、要求メッセージに対する応答を妨げられることのないマルチスレッド・サーバを得ることを目的とする。

【００１７】請求項６の発明は、スタック領域の不足を理由に、要求メッセージに対する処理が遅延するのを防止できるマルチスレッド・サーバを得ることを目的とする。

【００１８】請求項７の発明は、並列処理の効率を向上させ、高速に処理を行うことができるマルチスレッド・サーバを得ることを目的とする。

【００１９】請求項８の発明は、周期的に処理を行う周期起動型スレッドのリアルタイム性を確保できるマルチスレッド・サーバを得ることを目的とする。

【００２０】請求項９及び請求項１０の発明は、タイマスレッドの待ち時間をできるだけ短くできるマルチスレッド・サーバを得ることを目的とする。

【００２１】

【課題を解決するための手段】請求項１の発明に係るマルチスレッド・サーバは、第２の受信部により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を第１の受信部が受信してメモリに格納したサンプルデータに基づいて行うようにしたものである。

【００２２】請求項２の発明に係るマルチスレッド・サーバは、処理実行部の処理結果を一時的に保持するとともに、複数の処理結果を保持した場合には優先度の高い処理結果から順次外部装置に対して送信するようにしたものである。

【００２３】請求項３の発明に係るマルチスレッド・サーバは、処理実行部の処理結果を外部装置に対して送信する処理を行う際に、他の処理と競合する場合、該他の処理より優先度が高いときのみ処理を行うようにしたものである。

【００２４】請求項４の発明に係るマルチスレッド・サーバは、第２の受信部が複数の要求メッセージを保持している場合、優先度の高い要求メッセージから順次取り出すようにしたものである。

【００２５】請求項５の発明に係るマルチスレッド・サーバは、各スレッドが処理を実行する際に必要となるメモリ領域を各スレッド毎に予め確保しておき、各スレッドに処理を実行させる際に対応するメモリ領域を割り当てるようにしたものである。

【００２６】請求項６の発明に係るマルチスレッド・サーバは、処理に必要となるスレッドを予め生成し、待機状態にしておくようにしたものである。

【００２７】請求項７の発明に係るマルチスレッド・サーバは、第１の受信部により受信されたサンプルデータをメモリに格納中に、処理実行部から所定のアドレスに格納されているサンプルデータの読み込み要求があった場合、その読み込み要求に係るアドレスが、現在サンプルデータを格納しているアドレスと一致するときのみその読み込み要求を却下するようにしたものである。

【００２８】請求項８の発明に係るマルチスレッド・サーバは、所定の起動周期毎に周期起動型スレッドを起動させるとともに、その周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させるタイマスレッドを設けたものである。

【００２９】請求項９の発明に係るマルチスレッド・サーバは、タイマスレッドより優先度の低いスレッドがセマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能をそのセマフォに持たせたものである。

【００３０】請求項１０の発明に係るマルチスレッド・サーバは、共有資源を使用しているか否かを示すフラグ及びそのフラグを読み書きするためのセマフォを各スレッド毎に設けるとともに、タイマスレッドより優先度の低いスレッドが当該セマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能をそのセマフォに持たせたものである。

【００３１】

【作用】請求項１の発明におけるマルチスレッド・サーバは、要求メッセージを保持する第２の受信部を設けたことにより、要求メッセージの欠損が防止され、また、第２の受信部により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を第１の受信部が受信してメモリに格納したサンプルデータに基づいて行う処理実行部を設けたことにより、第２の受信部から要求メッセージを受信中でも、第１の受信部からサンプルデータを受信してメモリに格納することができる。

【００３２】請求項２の発明におけるマルチスレッド・サーバは、処理実行部の処理結果を一時的に保持するとともに、複数の処理結果を保持した場合には優先度の高い処理結果から順次外部装置に対して送信する送信部を設けたことにより、高速性の要求される処理結果から順次送信できるようになる。

【００３３】請求項３の発明におけるマルチスレッド・サーバは、処理実行部の処理結果を外部装置に対して送信する処理を行う際に、他の処理と競合する場合、該他の処理より優先度が高いときのみ処理を行う送信部を設けたことにより、処理結果を外部装置へ送信する処理よりも、高速性の要求される処理がある場合には、その高速性の要求される処理から優先して処理される。

【００３４】請求項４の発明におけるマルチスレッド・サーバは、第２の受信部が複数の要求メッセージを保持

している場合、優先度の高い要求メッセージから順次取り出す処理実行部を設けたことにより、要求メッセージの欠損を防止され、また、高速性の要求される要求メッセージに対する処理から行なえる。

【００３５】請求項５の発明におけるマルチスレッド・サーバは、各スレッドが処理を実行する際に必要となるメモリ領域を各スレッド毎に予め確保しておき、各スレッドに処理を実行させる際に対応するメモリ領域を割り当てる処理実行部を設けたことにより、ガベージコレクションを行う必要がなくなる。

【００３６】請求項６の発明におけるマルチスレッド・サーバは、処理に必要となるスレッドを予め生成し、待機状態にしておく処理実行部を設けたことにより、スタック領域の不足を理由に、要求メッセージに対する処理が遅延するのを防止される。

【００３７】請求項７の発明におけるマルチスレッド・サーバは、第１の受信部により受信されたサンプルデータをメモリに格納中に、処理実行部から所定のアドレスに格納されているサンプルデータの読み込み要求があった場合、その読み込み要求に係るアドレスが、現在サンプルデータを格納しているアドレスと一致するときのみその読み込み要求を却下するようにしたことにより、同じデータを同時に読み書きすることによって生じるデータ内容の矛盾を防止しつつデータの読み書きの並列処理が可能になる。

【００３８】請求項８の発明におけるマルチスレッド・サーバは、所定の起動周期毎に周期起動型スレッドを起動させるとともに、その周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させるタイマスレッドを設けたことにより、他の周期起動型スレッドを所定の起動周期毎に確実に起動できるようになる。

【００３９】請求項９の発明におけるマルチスレッド・サーバは、タイマスレッドより優先度の低いスレッドがセマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能をそのセマフォに持たせたことにより、優先度の低いスレッドがセマフォを獲得していても、その優先度の低いスレッドの処理が速やかに行われるため、タイマスレッドがセマフォを獲得するまでの待ち時間が短縮される。

【００４０】請求項１０の発明におけるマルチスレッド・サーバは、共有資源を使用しているか否かを示すフラグ及びそのフラグを読み書きするためのセマフォを各スレッド毎に設けるとともに、タイマスレッドより優先度の低いスレッドが当該セマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能をそのセマフォに持たせたことにより、優先度の低いスレッドがセマフォを獲得していても、その優先度の低いスレッドの処理が速やかに行われるため、タイマスレッドがセマフォを獲得するまでの待ち時間が短縮される。

【００４１】

【実施例】

実施例１．以下、この発明の一実施例を図について説明する。図１はこの発明の実施例１によるマルチスレッド・サーバを示す構成図であり、図において、従来のものと同一符号は同一または相当部分を示すので説明を省略する。１１はプラント１からサンプルデータを受信するＦＩＦＯキュー（第１の受信部）、１２はＦＩＦＯキュー１１が受信したサンプルデータを一定周期でリングバッファ１３に書き込む周期起動型スレッド、１３はそのサンプルデータを格納するリングバッファ（メモリ）、１４は周期起動型スレッド１２が現在書き込み中であるリングバッファ１３のインデックス（アドレス）より少し過去のサンプルデータをいくつかまとめてディスク１５へ一定周期で書き込む周期起動型スレッド、１５はディスク（メモリ）である。

【００４２】また、１６はクライアント２からネットワーク３を介して要求メッセージを受信するレシーバ、１７はレシーバ１６が受信したその要求メッセージを保持するＦＩＦＯキューであり、レシーバ１６及びＦＩＦＯキュー１７から第２の受信部が構成されている。

【００４３】また、１８はＦＩＦＯキュー１７により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じてチャイルドスレッド６，７，８のいずれかを起動することにより、その要求メッセージの内容に応じた処理をリングバッファ１３及びディスク１５に格納されたサンプルデータに基づいて行うマザースレッドであり、マザースレッド１８及びチャイルドスレッド６，７，８から処理実行部が構成されている。

【００４４】また、１９はチャイルドスレッド６，７，８の処理結果を一時的に保持するＦＩＦＯキュー、２０はＦＩＦＯキュー１９により保持されたその処理結果をネットワーク３を介してクライアント２に対して送信するトランスミッタであり、ＦＩＦＯキュー１９及びトランスミッタ２０から送信部が構成されている。

【００４５】次に動作について説明する。まず、サンプルデータは、例えば、１０ｍｓｅｃ毎にプラント１から送信され、ＦＩＦＯキュー１１がこれを受信する。そして、ＦＩＦＯキュー１１に受信されたサンプルデータは、周期起動型スレッド１２によって周期的にリングバッファ１３に格納される。また、リングバッファ１３にある程度サンプルデータが蓄積されていくと、周期起動型スレッド１４によってサンプルデータをいくつかまとめてディスク１５に格納される。なお、リングバッファ１３及びディスク１５にサンプルデータを格納する際、いつサンプルされたサンプルデータであるか分かるよう

に時刻情報もいっしょに格納される。

【００４６】一方、要求メッセージは、不定期にクライアント２からネットワーク３を介して送信され、レシーバ１６がこれを受信する。そして、レシーバ１６に受信された要求メッセージは、一時的にＦＩＦＯキュー１７に格納される。そして、ＦＩＦＯキュー１７に格納された要求メッセージを、マザースレッド１８が順次取り込み、その要求メッセージの内容に応じてチャイルドスレッド６，７，８のいずれかを生成して起動する（チャイルドスレッド６，７，８を起動する際、処理に必要なメモリ領域を動的に確保する。詳細は後述する。）。ここで、要求メッセージには、下記に示すようなものがある。

① あるイベントが起こった場合にそのイベントの発生を通知

② 現在時刻のサンプルデータの周期転送

③ 過去のある時刻からある時刻までのサンプルデータの一斉転送

【００４７】そして、マザースレッド１８により起動されたチャイルドスレッド６，７，８は、要求メッセージの内容に応じた処理をリングバッファ１３及びディスク１５に格納されているサンプルデータに基づいて行う（必要とするサンプルデータがリングバッファ１３に格納されていない場合、ディスク１５とアクセスする）。例えば、要求メッセージの内容が、上記①の内容であれば、イベントの起動があった旨の通知をクライアント２に対して送信する処理を行う。また、要求メッセージの内容が、上記②③の内容であれば、リングバッファ１３等から該当するサンプルデータを読み込んでクライアント２に対して送信する処理を行う。

【００４８】そして、チャイルドスレッド６，７，８の処理結果はＦＩＦＯキュー１９に送信され、一時的に保持される。なお、チャイルドスレッド６，７，８は並列処理が可能であるため、ＦＩＦＯキュー１９に複数の処理結果が保持されることがある。そして、ＦＩＦＯキュー１９に保持された処理結果は、トランスミッタ２０が順次取り込んでネットワーク３を介してクライアント２に対して送信し、一連の処理を終了する。

【００４９】このように、実施例１では、要求メッセージを受信するレシーバ１６の他に、プラント１から送信されるサンプルデータを受信する専用のＦＩＦＯキュー１１を設けるようにしているので、要求メッセージの処理と別個独立にサンプルデータのサンプル処理を行うことができるようになり、従って、要求メッセージの処理によってサンプル処理が妨げられる事態が発生することがなくなり、サンプル処理をその時間上の制約を保って行うことができる。また、要求メッセージを一時的に保持するＦＩＦＯキュー１７を設けているので、前に送られた要求メッセージに対する処理が終了する前に他の要求メッセージが複数個クライアント２から送信されてきたとしても、要求メッセージが欠損するという事態は発生しない。因に、この実施例１は請求項１の発明に対応している。

【００５０】実施例２．上記実施例１では、ＦＩＦＯキュー１９に複数の処理結果を保持された場合、ＦＩＦＯキュー１９に保持された順にその処理結果を送信するものについて説明したが、図２に示すように、複数の処理結果を保持した場合には優先度の高い処理結果から順次クライアント２に対して送信するようにしてもよい。図２において、１９ａ、１９ｂ、１９ｃはＦＩＦＯキュー１９の構成要素であり、それぞれチャイルドスレッド６，７，８の処理結果を一時的に保持する。また、２１はＦＩＦＯキュー１９が複数の処理結果を保持した場合には優先度の高い処理結果から順次クライアント２に対して送信するトランスミッタ（送信部）である。

【００５１】次に動作について説明する。まず、要求メッセージは、上記のごとく内容がそれぞれ異なっており、従って、それぞれクライアント２に対して応答するに際して時間上の制約に関係した優先度が存在する。例えば、チャイルドスレッド６は、上記①の要求メッセージに対する処理を行うものである場合、かかる処理は、イベント発生に対するクライアント２の対応が間に合うように、ある時間以内にクライアント２に送信される必要があるので極めて高い優先度P_{mq2}を有している。また、チャイルドスレッド７は、上記②の要求メッセージに対する処理を行うものである場合、かかる処理は、クライアント２がまだ最新のサンプルデータとして利用する価値のある時間以内に送信される必要があるので、チャイルドスレッド６の処理に次いで高い優先度P_{mq1}を有する。また、チャイルドスレッド８は、上記③の要求メッセージに対する処理を行うものである場合、かかる処理は、送信が多少遅れても支障が少ないのでチャイルドスレッド６，７の処理より低い優先度P_{mq0}を有している。

$P_{mq0} < P_{mq1} < P_{mq2}$

【００５２】この場合において、チャイルドスレッド８から優先度P_{mq0}の処理結果がＦＩＦＯキュー１９ｃに送信されて格納された後、直ちにチャイルドスレッド６から優先度P_{mq2}の処理結果がＦＩＦＯキュー１９ａに送信されて格納されたとする。実施例１では、ＦＩＦＯキュー１９に格納された順に送信するので、チャイルドスレッド８の処理結果を先に送信することになるが、実施例２では、処理結果の優先度を考慮して送信するので、優先度の高いチャイルドスレッド６の処理結果を先に送信することになる。これにより、クライアント２の要求に則した処理が可能になる。因に、この実施例２は請求項２の発明に対応している。

【００５３】実施例３．上記実施例２では、トランスミッタ２１は、送信以外の他の処理を行うスレッド２２の存在を確認せずに処理結果を送信するものについて説明

したが、図３に示すように、チャイルドスレッド６，７，８の処理結果をクライアント２に対して送信する処理を行う際に、送信以外の他の処理を行うスレッド２２と競合する場合、スレッド２２の処理の優先度P_{etc}より高い優先度をトランスミッタ２１の送信スレッドが有するときのみ送信処理を行うようにしてもよい。図において、２１ａ、２１ｂ、２１ｃはトランスミッタ２１の送信スレッド、２２は送信以外の他の処理を行うスレッド、２３はセマフォである。

【００５４】次に動作について説明する。例えば、送信スレッド２１ａ，２１ｂ，２１ｃの優先度をそれぞれP_{tr2}，P_{tr1}，P_{tr0}とし、優先度の高さの関係は下記のようにあるとする。

$P_{tr0} < P_{tr1} < P_{etc} < P_{tr2}$

【００５５】この場合において、送信スレッド２１ａが送信処理を行うときは、送信スレッド２１ａの優先度P_{tr2}は、スレッド２２の優先度P_{etc}より高いので、スレッド２２が存在していても直ちに送信処理が行われる。一方、送信スレッド２１ｃが送信処理を行うときは、送信スレッド２１ｃの優先度P_{tr0}は、スレッド２２の優先度P_{etc}より低いので、スレッド２２の処理が終了するまで送信処理が待機される。

【００５６】なお、セマフォ２３は、複数の送信スレッド２１ａ，２１ｂ，２１ｃが同時にネットワーク３へ送信するのを避けるために設けられている。セマフォ２３を獲得できる送信スレッドは１つであり、セマフォ２３を獲得している送信スレッド（図３の例では送信スレッド２１ａが獲得している）のみがネットワーク３に対して送信することができる。

【００５７】ここで、セマフォ２３を設けた場合下記の不具合がある。即ち、低い優先度を有する送信スレッド２１ｃがセマフォ２３を獲得した後に、高い優先度を有する送信スレッド２１ａがＦＩＦＯキュー１９ａに格納されたような場合、低い優先度を有する送信スレッド２１ｃがセマフォ２３を獲得しているので、高い優先度を有する送信スレッド２１ａは送信スレッド２１ｃの送信処理を待たなければならいが、送信スレッド２１ｃの優先度はスレッド２２の優先度より低いので、送信スレッド２１ｃはスレッド２２の処理が終了するまで待機状態になる。従って、スレッド２１ａは高い優先度を有しているのにもかかわらず、結果として、スレッド２２及び送信スレッド２１ｃの処理が終了するまで送信処理を行うことができない不具合がある。

【００５８】そこで、セマフォ２３に優先度継承機能（送信スレッドの優先度変更機能）を持たせることにより上記不具合を解消することができる。即ち、一時的に送信スレッド２１ｃの優先度を送信スレッド２１ａの優先度まで高めることにより、送信スレッド２１ｃの送信処理をスレッド２２に優先させる。送信スレッド２１ｃの処理が終了すれば、送信スレッド２１ａがセマフォ２３を獲得し、送信スレッド２１ａの優先度はスレッド２２の優先度より高いので、送信スレッド２１ａはスレッド２２の処理を待たずに送信が可能になる。従って、送信スレッド２１ａの待ち時間を短縮することができる。因に、この実施例３は請求項３の発明に対応している。

【００５９】実施例４．上記実施例１では、要求メッセージがＦＩＦＯキュー１７に保持された順に、マザースレッド１８が取り出して処理するものについて説明したが、図４に示すように、要求メッセージに優先度を持たせ、ＦＩＦＯキュー１７に先に格納された要求メッセージの優先度より、後に格納された要求メッセージの優先度が高い場合、後に格納された要求メッセージからマザースレッド１８が取り込んで処理するようにしてもよい。これにより、高速性の要求される要求メッセージに対する処理から行うことができるようになる。因に、この実施例４は請求項４の発明に対応している。

【００６０】実施例５．上記実施例１では、要求メッセージの内容に応じてマザースレッド１８がチャイルドスレッド６，７，８のいずれかを生成して起動する際、処理に必要なメモリ領域を動的に確保するものについて説明したが、図５に示すように、各チャイルドスレッド６，７，８が処理を実行する際に必要となるメモリ領域を各スレッド６，７，８毎に予め確保しておき、各スレッド６，７，８に処理を実行させる際に対応するメモリ領域を割り当てるようにしてもよい。また、上記実施例１では、マザースレッド１８がその都度チャイルドスレッド６，７，８を生成するようにしていたが、チャイルドスレッド６，７，８を予め生成して待機状態にしておいてもよい。

【００６１】以下、図５について詳細に説明する。図５において、３１はスレッドプール、３２，３３，３４はそれぞれチャイルドスレッド６，７，８が処理を行う際に必要となるメモリ領域であり、スレッドを実行すべき関数へのポインタ、その関数への引数、スレッドの優先度、周期起動型スレッドの場合には、周期及びデッドライン、イベント起動型スレッドの場合には、待つべきイベント（以下、起動イベントという）の種類などの情報が格納される。なお、各チャイルドスレッド６，７，８は初期化時には待機状態にある。

【００６２】また、３５はスレッドプール３１を管理するスレッドプール管理テーブル、３６は管理テーブルセマフォ、３７は待機中のチャイルドスレッド６，７，８の数を格納する領域、３８はマザースレッド１８の起動要求をスレッドプール３１に対して知らせる要求セマフォ、３９は起動要求を受けつけたチャイルドスレッド６，７，８のスレッドＩＤを格納する領域、４０はスレッドプール３１からマザースレッド１８へ要求の受領を知らせるための受領セマフォ、４１はマザースレッド１８からスレッドプール３１へ起動命令を伝えるための起動セマフォである。また、４２はマザースレッド１８以

外にスレッドプール３１を用いるスレッドである。

【００６３】次に動作について説明する。まず、マザースレッド１８は、スレッドプール管理テーブル３５へのアクセス権を得るために管理テーブルセマフォ３６にロックをかける（動作1）。そして、管理テーブルセマフォ３６へのロックに失敗した場合には、一定時間後に再度ロックを試みる。管理テーブルセマフォ３６へのロックに成功した場合には、領域３７に格納されている待機中のスレッドを確認し（動作２）、“０”であれば起動不可能として例外処理を行うか、あるいは、一定時間後に再度起動を試みる。“１”以上であれば要求セマフォ３８に対してシグナルを送信する（動作３）。

【００６４】また、待機中のチャイルドスレッド６，８はすべて要求セマフォ３８からのシグナルの到着を待っており、マザースレッド１８からのシグナルは、これらの待機中のチャイルドスレッド６，８のいずれか１つによって受信される。この例の場合では、チャイルドスレッド６によって受信されたものとする。

【００６５】チャイルドスレッド６は、起動を受領したスレッドが自分であることをマザースレッド１８に知らせるため、自分のスレッドＩＤを領域３９に書き込み（動作５）、その後、受領セマフォ４０にシグナルを送信し（動作６）、起動セマフォ４１へのシグナルの到着を待つ。

【００６６】一方、マザースレッド１８は、動作３の終了後、受領セマフォ４０へのシグナルの到着を待っており、チャイルドスレッド６からのシグナルを受信すると（動作７）、領域３９から要求を受領したスレッドＩＤを読み取る（動作８）。

【００６７】ここで、マザースレッド１８は、予め確保されているメモリ領域３２に対して起動に必要な情報を書き込み（動作９）、書き込みが完了すると、起動セマフォ４１へシグナルを送信する（動作１０）。チャイルドスレッド６は、起動セマフォ４１からシグナルを受信すると（動作１１）、領域３７の待機中のスレッドの数を“１”減らし（動作１２）、管理テーブルセマフォ３６へのロックを解除する（動作１３）。

【００６８】このように、予め各チャイルドスレッド６，７，８が実行時に必要となるメモリ領域３２，３３，３４を確保しておくことにより、メモリ領域確保の繰り返しを原因とするメモリ空間上の不要なメモリが増加していくという事態は発生しなくなる。従って、ガベージコレクションを行う必要もなくなる。また、予め各チャイルドスレッド６，７，８を初期化時に生成しておくことにより、スタック領域の不足を原因とするチャイルドスレッド６，７，８の生成不可という事態は発生しなくなる。従って、スタック領域が解放されるまで処理を待機しなければならないという従来の問題点は解消される。因に、この実施例５は請求項５及び請求項６の発明に対応している。

【００６９】次に、リアルタイム性を要求されないチャイルドスレッドのスレッドプール３１内の構成及び動作について説明する。図６はスレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。実行すべき関数へのポインタ、その関数への起動要求をだしたマザースレッド１８からの引数、実行の優先度が含まれている。また、図７はスレッドプール３１内のチャイルドスレッドの動作を示すフローチャートであり、マザースレッド１８、あるいは他のスレッド４２から終了の割り込みがかかると、ステップＳＴ１１に制御を移す。

【００７０】次に、周期起動型スレッドのスレッドプール３１内の構成及び動作について説明する。図８はスレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。一周期ごとに実行すべき関数へのポインタ、その関数への起動要求をだしたマザースレッド１８からの引数、実行の優先度、起動の周期、一周期あたりの処理のデッドラインが含まれている。また、図９はスレッドプール３１内のチャイルドスレッドの動作を示すフローチャートであり、マザースレッド１８、あるいは他のスレッド４２が実行中の周期起動型スレッドを停止させる場合には、各周期起動型スレッドの持つ終了フラグを立てる。各周期起動型スレッドはその一周期ごとに終了フラグをチェックし、それが立っていれば処理を終了し、待機状態に戻る。

【００７１】次に、イベント起動型スレッドのスレッドプール３１内の構成及び動作について説明する。図１０はスレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。イベントが発生したときに実行すべき関数へのポインタ、その関数への起動要求をだしたマザースレッド１８からの引数、実行の優先度、起動イベントに関する情報、処理のデッドラインが含まれている。また、図１１はスレッドプール３１内のチャイルドスレッドの動作を示すフローチャートである。マザースレッド１８、あるいは他のスレッド４２がイベント待ちをしているイベント起動型スレッドを待機状態にする場合には、そのスレッドに対して終了イベントを送信する。終了イベントを受信したスレッドは待機状態に戻る。

【００７２】スレッドプール３１内の各チャイルドスレッド６，７，８に、リアルタイム性を要求されないスレッド、周期起動型スレッド、イベント起動型スレッドのいずれの動作をも行わせるようにすることができる。マザースレッド１８からスレッドプール３１内の各チャイルドスレッド６，７，８へ受け渡す起動情報の中に起動するスレッドの種類を含ませる。スレッドプール３１内のチャイルドスレッド６，７，８はこの情報に基づき行う処理を決定する。また、図１２はスレッドプール３１内の各チャイルドスレッド６，７，８がもつ起動情報の構成図であり、図１３はスレッドプール３１内の各チャイルドスレッドの動作を示すフローチャートである。

【００７３】ここで、スレッドプール３１内の数をｎとする。クライアント２から送られるスレッドの起動要求（要求メッセージ）によってこれらのスレッドに処理が受け渡される。あるイベントが生じた場合の緊急通知を行うスレッドの数の上限をn_1、現在起動されている数をi_1、サンプルデータの周期的な転送を行うスレッドの数の上限をn_2、現在起動されている数をi_2、あるイベントが生じた場合の任意期間のサンプルデータの転送を行うスレッドの数の上限をn_3、現在起動されている数をi_3とする。また、下記の関係があるとする。

$n_1 + n_2 + n_3 > n$

$n_1 \leqq n$

$n_2 \leqq n$

$n_3 \leqq n$

【００７４】この場合において、i_1，i_2が小さく、下記の関係がある場合には、任意期間のサンプルデータの転送を行うスレッドを最大数n_3まで起動することができる。

$i_1 + i_2 + i_3 \leqq n$　　・・・（１）

そして、緊急通知を行うスレッドや周期転送を行うスレッドの起動要求がクライアント２から送られて式（１）の関係を満たさなくなった場合には、これを満たすように任意期間のサンプルデータの転送を行うスレッドのいくつかを中断あるいは終了させる。

【００７５】実施例6．図１４はリングバッファ１３及びディスク１５に対して複数のスレッドが書き込み及び読み出しを行う動作を説明する構成図である。以下、リングバッファ１３等をアクセスするスレッドの機能に応じて説明する。

【００７６】①　書き込みスレッド

まず、周期起動型スレッド１２（書き込みスレッド）は、リングバッファ１３へ周期ごとに書き込みを行う。書き込まれるリングバッファ１３のインデックスは書き込みごとに一つずつ移動する。ここで、周期起動型スレッド１２が現在書き込みを行っているインデックス（図中、×で示してあるインデックス）を格納する領域５１が予め用意されている。また、領域５１は複数のスレッドからアクセスされるため、領域５１の相互排除のためセマフォ５２が設けられている。

【００７７】そして、周期起動型スレッド１２は書き込み中に、他のスレッドが書き込み中のインデックスにアクセスするのを防止すべく、書き込むインデックスを変えるごとに、領域５１にそのインデックスを書き込む。なお、各スレッドの領域５１へのアクセスに要する時間は小さく、またセマフォ５２に優先度承継機能を持たせているので、各スレッドがセマフォ５２の獲得のために長い時間待たされることはない。

【００７８】②　バックアップスレッド

周期起動型スレッド１４（バックアップスレッド）はリングバッファ１３上のサンプルデータを、それらが失われないうちにディスク１５へ格納する。ディスク１５上にはバックアップ用のファイルがいくつか用意され、各スレッドからは図に示すように、リング状に配置されているように見ることができる。周期起動型スレッド１４は、リングバッファ１３上に対して周期起動型スレッド１２が書き込んでいるインデックスのサンプルデータよりある程度遅れた時刻のサンプルデータを一度にいくつかまとめてディスクに転送する。

【００７９】周期スレッド１２及び周期スレッド１４がアクセスするリングバッファ１３上のインデックスの移動の速度は等しく、また、周期スレッド１４は周期スレッド１２が書き込みを行っているインデックスから十分はなれたインデックスを読み出すので、周期スレッド１２及び周期スレッド１４が同じインデックスを同時にアクセスすることはない。また、ディスク１５上のファイルには、各ファイルに定められた容量まで書き込み、その容量に達するとそのファイルをクローズし、次のファイルへの書き込みを始める。

【００８０】ここで、ディスク１５に対しても、現在書き込みを行っているファイルに関する領域５３を用意するとともに、領域５３の相互排除のためセマフォ５４を設けている。領域５３の格納する情報は、現在書き込みが行われているファイルの番号とそのファイル上にある最も過去のサンプルデータの時刻である。従って、周期起動型スレッド１４は、現在書き込みを行っているファイルの番号を領域５３に書き込むようにしている。また、新しいファイルに書き込みを始める際には、そのファイルに書き込んだ最初のサンプルデータの時刻を領域５３に書き込むようにしている。

【００８１】③　周期的な読み出しを行うスレッド

例えば、チャイルドスレッド７が周期起動型スレッドの場合、チャイルドスレッド７は、一周期ごとに、リングバッファ１３上に格納されている最新のサンプルデータを読み出す。チャイルドスレッド７はその周期ごとに領域５１の内容を読み出し、現在周期起動型スレッド１２が書き込みを行っているインデックスの１つ手前のインデックス（図中、○で示してあるインデックス）の内容を読み出す。

【００８２】④　任意期間の読み出しを行うスレッド

例えば、チャイルドスレッド８が任意期間の読み出しを行うスレッドの場合、チャイルドスレッド８は、任意時刻から任意時刻までのサンプルデータを任意の間隔で読み出す。要求されたサンプルデータのうちリングバッファ１３及びディスク１５上にないものがある場合には、リングバッファ１３及びディスク１５上にあるサンプルデータのみを読み出しに先行して、そのサンプルデータをリングバッファ１３から読み出すべきか、ディスク１５から読み出すべきかを決定する。

【００８３】領域５３に格納されている現在書き込まれているファイル上の最も過去のサンプルデータの時刻

と、読み出すべきサンプルデータの時刻を比較する。後者の方が過去のものであれば、既にディスク１５上から失われているほどの過去のサンプルデータでない限り、ディスク１５から読み出す。後者が前者と同じ、あるいは後者の方が前者より新しい時刻であれば、領域５１の示すインデックスに格納されているサンプルデータより新しい時刻でない限り、リングバッファ１３から読み出す（図中、△で示してあるインデックス）。

【００８４】このように、実施例６によれば、ＦＩＦＯキュー１１により受信されたサンプルデータをリングバッファ１３に格納中に、チャイルドスレッド７，８から所定のアドレスに格納されているサンプルデータの読み込み要求があった場合、現在サンプルデータを格納しているアドレスよりも手間のアドレスの内容に限り読み出せるようにしているので、即ち、読み込み要求に係るアドレスが現在サンプルデータを格納しているアドレスと一致するときはその読み込み要求を却下するようにしているので、同じデータを同時に読み書きすることによって生じるデータ内容の矛盾を防止しつつデータの読み書きの並列処理が可能になり、その結果、高速に処理を行うことができる効果がある。因に、この実施例６は請求項７の発明に対応している。

【００８５】実施例７．上記実施例１～６では、周期起動型スレッドの処理の終了については特に言及しなかったが、周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させるようにしてもよい。これにより、上記実施例１～６であれば、ある周期起動型スレッドの処理が複雑である等のため処理時間内に処理が終了しない場合には、リアルタイム性を要求される他の周期起動型スレッドの処理の開始が遅れる等の不具合があったが、他の周期起動型スレッドを所定の起動周期毎に確実に起動できるようになる。

【００８６】図１５はこの発明の実施例７によるマルチスレッド・サーバを示す構成図であり、図において、６１はマザースレッド１８が生成するチャイルドスレッド６，７，８等のうち、周期的に起動される周期起動型スレッド７，１２，１４の起動及び終了を管理するタイマスレッドであり、そのタイマスレッド６１は所定の起動周期毎にその周期起動型スレッド７，１２，１４を起動させるとともに、その周期起動型スレッド７，１２，１４が起動後、所定時刻（デッドライン）経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させる機能を有している。６２は各スレッドの情報を格納するスレッド管理テーブル（資源）である（図１６参照）。

【００８７】次に動作について説明する。まず、スレッド管理テーブル６２には、図１６に示すように、当該スレッドが周期起動型スレッドであるか否かを示すUフラグ、当該スレッドが周期毎の処理を実行中であるか否かを示すＥフラグ、当該スレッドの起動時刻までの時間をタイマスレッド６１の起動回数（例えば、タイマスレッド６１の起動周期が１００ｍｓであって、スレッドの起動時刻までの時間が２００ｍｓであれば、起動回数は２回となる）で示すＵＳフィールド（既に起動されている場合には負の値になる）、当該スレッドの次の起動までの時間をタイマスレッド６１の起動回数で示すUWフィールド、当該スレッドのデッドラインまでの時間をタイマスレッド６１の起動回数示すUDフィールド、当該スレッドの周期を格納する周期フィールド、当該スレッドのデッドラインを格納するデッドラインフィールドが記憶されている。

【００８８】そして、スレッド管理テーブル６２に記憶されている各スレッドの情報は、マザースレッド１８が管理しており、具体的には、マザータスク１８が要求メッセージにしたがってチャイルドスレッドを生成等するときに、そのチャイルドスレッドの情報をスレッド管理テーブル６２に登録する。一方、当該スレッドの消滅、停止要求等を受けるとそのスレッドの登録を抹消する。

【００８９】従って、スレッド管理テーブル６２の記憶内容を解読すれば、どのスレッドが周期起動型のスレッドであるか否か、当該スレッドの起動周期、当該スレッドのデッドライン等の情報を得ることができることになるので、タイマスレッド６１は、周期起動型スレッドの起動周期に比べて極めて短い周期毎にスレッド管理テーブル６２の記憶内容を解読する。

【００９０】そして、タイマスレッド６１は、ある周期起動型スレッドが起動時刻に到達していれば、その周期起動型スレッドに対して起動シグナルを通知することにより、その周期起動型スレッドを起動させる。また、その周期起動型スレッドが起動してから所定時刻経過しても処理が終了しない場合には、その周期起動型スレッドにタイムアウトを通知することにより、その周期起動型スレッドの処理を強制的に中断させるとともに、他の周期起動型スレッドの処理を妨げないために、その周期起動型スレッドが獲得している資源を解放させるなどの必要なタイムアウト処理を実行させる。

【００９１】このように、実施例７によれば、タイマスレッド６１が、周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させるようにしているので、ある周期起動型スレッドの処理が複雑である等のため処理時間内に処理が終了しない場合でも、他の周期起動型スレッドの起動に影響が及ぶことがなく、周期起動型スレッドのリアルタイム性を確保することができる。因に、この実施例７は請求項８の発明に対応している。

【００９２】実施例８．上記実施例７では、各スレッド間の共有資源であるスレッド管理テーブル６２に対する読み書きの相互排除について特に言及しなかったが、スレッド管理テーブル６２は各スレッド間の共有資源であ

るため、複数のスレッドが同一のデータを同時に読み書きする場合が考えられる。そして、同一データの同時読み書きがなされた場合には、その影響でデータ内容に矛盾が生じる場合があるので、読み書きに対して相互排除の必要性がある。例えば、タイマスレッド６１がスレッド管理テーブル６２の内容を読み込み中に、周期起動型スレッド等がスレッド管理テーブル６２の内容を書き換えてしまうと、タイマスレッド６１は正しいデータが得られなくなる場合がある。

【００９３】そこで、相互排除の方法として、スレッドがセマフォを獲得した場合に限りスレッド管理テーブル６２に対して読み書きできるようにする手法が考えられるが、単にセマフォの獲得だけを読み書きできる条件とすると、例えば下記に示すような不具合を生じる場合がある。即ち、タイマスレッド６１より優先度の低いスレッド７等がセマフォを獲得しているためにそのタイマスレッド６１が待ち状態になっている場合、タイマスレッド６１がセマフォを獲得して起動するためには、その優先度の低いスレッド７等が処理を終了してセマフォを解放しなければならないが、そのスレッド７等は優先度が低いために、他のスレッドの処理が終了するのを待つ待機状態になる場合があり、この場合、スレッド７等は処理を終了することができないためセマフォを解放することができず、その結果、タイマスレッド６１が起動できないという不具合を生じることがある。因に、タイマスレッド６１は、他のスレッドの起動等を管理するスレッドであるため、マザースレッド１８や周期起動型スレッド等よりも優先度が高く設定されている。

【００９４】この実施例８は、上記のような不具合を解消しつつスレッド管理テーブル６２に対する読み書きの相互排除を行うものであり、実施例８の構成を下記に示す。即ち、この実施例８では、図１６に示すように、スレッド管理テーブル６２に対する読み書きの相互排除を行うためにセマフォ６３，６４を設けるとともに、タイマスレッド６１より優先度の低いスレッド７等がそのセマフォ６３，６４を獲得しているためにそのタイマスレッド６１が待ち状態になっている場合、一時的にその優先度の低いスレッド７等の優先度をそのタイマスレッド６１の優先度と同一にする優先度継承機能をセマフォ６３，６４に持たせるようにしたものである。

【００９５】これにより、本来優先度の低いスレッド７等の優先度が、優先度の高いタイマスレッド６１の優先度と一時的に同一になるため、優先度の関係で他のスレッドの処理の実行に伴ってスレッド７等が待機状態になる場合がほとんどなくなり、その結果、スレッド７等の処理が速やかに行われてセマフォ６３，６４が解放されるので、タイマスレッド６１は速やかにセマフォ６３，６４を獲得できるようになる。従って、周期起動型スレッドのリアルタイム性を確保できるとともに、装置の信頼性が向上する。

【００９６】ここで、図１７はタイマスレッド６１の動作を示すフローチャートであり、その動作を簡単に説明すると、タイマスレッド６１は、周期起動型スレッドの起動時刻、周期、デッドラインを表す時間の粒度（最小単位）を周期として起動される。例えば、起動時刻、周期、デッドラインがすべて１００ｍｓ単位で表されるものであれば、タイマスレッド６１は、１００ｍｓ周期で起動する。そして、タイマスレッド６１は各周期毎に起動され、ステップＳＴ５１でＵフラグを読み取るためにセマフォ６３を獲得し、ステップＳＴ６７及びステップＳＴ７２でＥフラグを読み取るためのセマフォ６４を獲得する。また、ステップＳＴ５３でセマフォ６３を解放し、ステップＳＴ６９及びステップＳＴ７４でセマフォ６４を解放する。

【００９７】次に、図１８はマザースレッド１８がスレッド管理テーブル６２に周期起動型スレッドを登録する動作を示すフローチャートであり、その動作を簡単に説明すると、ステップＳＴ８１でセマフォ６３を獲得したのちステップＳＴ８２でＵフラグの読み取りを行い、ステップＳＴ８３でセマフォ６３を解放する（以下、ステップＳＴ８１～８３の間をクリティカルセクションＵ１という）。また、ステップＳＴ８６でセマフォ６３を獲得したのちステップＳＴ８７でＵフラグの書き込み読みを行い、ステップＳＴ８８でセマフォ６３を解放する（以下、ステップＳＴ８６～８８の間をクリティカルセクションＵ２という）。

【００９８】次に、図１９はマザースレッド１８がスレッド管理テーブル６２から周期起動型スレッドの登録を削除する動作を示すフローチャートであり、その動作を簡単に説明すると、ステップＳＴ９１でセマフォ６３を獲得したのちステップＳＴ９２でＵフラグの書き込みを行い、ステップＳＴ９３でセマフォ６３を解放する（以下、ステップＳＴ９１～９３の間をクリティカルセクションＵ３という）。

【００９９】次に、図２０は各周期起動型スレッドの動作を示すフローチャートであり、その動作を簡単に説明すると、ステップＳＴ１０３でセマフォ６４を獲得したのちステップＳＴ１０４でＥフラグの書き込みを行い、ステップＳＴ１０５でセマフォ６４を解放する（以下、ステップＳＴ１０３～１０５の間をクリティカルセクションＥ１という）。また、ステップＳＴ１１１でセマフォ６４を獲得したのちステップＳＴ１１２でＥフラグの書き込み読みを行い、ステップＳＴ１１３でセマフォ６４を解放する（以下、ステップＳＴ１１１～１１３の間をクリティカルセクションＥ２という）。

【０１００】以上で明らかなように、タイマスレッド６１が起動時に、優先度の低いスレッド（マザースレッド１８、周期起動型スレッド）によって待たされる時間は、セマフォ６３，６４が優先度継承機能を有しているので、最も待たされる場合でも、クリティカルセクショ

ンＵ１，Ｕ２，Ｕ３のうちの何れか１つを実行するのに要する時間と、クリティカルセクションＥ１，Ｅ２のうちの何れか一方の実行に要する時間とを加えた時間に限られる。また、その加えた時間は、高々フラグ２個の読み書きに要する時間であって、タイマスレッド６１が行うすべての処理に比べて極めて短い時間であり、無視できる程度の時間である。因に、この実施例８は請求項９の発明に対応している。

【０１０１】実施例９．上記実施例８では、ＵフラグまたはＥフラグの全体にセマフォ６３，６４を設けたものについて示したが、図２１に示すように、スレッド管理テーブル６２を使用しているか否かを示すＳフラグ及びそのＳフラグを読み書きするためのセマフォ６５を各スレッド毎に設け、そして、タイマスレッド６１より優先度の低いスレッド７等が当該セマフォ６５を獲得しているためにそのタイマスレッド６１が待ち状態になっている場合、一時的にその優先度の低いスレッド７等の優先度をそのタイマスレッド６１の優先度と同一にする優先度継承機能をセマフォ６５に持たせるようにしてもよく、上記実施例８と同様の効果を奏する。因に、この実施例９は請求項１０の発明に対応している。

【０１０２】

【発明の効果】以上のように、請求項１の発明によれば、第２の受信部により保持された要求メッセージを順次取り出し、その要求メッセージの内容に応じた処理を第１の受信部が受信してメモリに格納したサンプルデータに基づいて行うように構成したので、要求メッセージを受信している途中であっても、サンプルデータを受信してメモリに格納できるようになり、その結果、サンプル処理をその時間上の制約を保って行うことができる効果がある。

【０１０３】請求項２の発明によれば、処理実行部の処理結果を一時的に保持するとともに、複数の処理結果を保持した場合には優先度の高い処理結果から順次外部装置に対して送信するように構成したので、高速性の要求される処理結果から順次送信できるようになり、その結果、外部装置の要求を十分満足できるようになる効果がある。

【０１０４】請求項３の発明によれば、処理実行部の処理結果を外部装置に対して送信する処理を行う際に、他の処理と競合する場合、該他の処理より優先度が高いときのみ処理を行うように構成したので、処理結果を外部装置へ送信する処理よりも、高速性の要求される処理がある場合には、その高速性の要求される処理を優先して処理ができるようになる効果がある。

【０１０５】請求項４の発明によれば、第２の受信部が複数の要求メッセージを保持している場合、優先度の高い要求メッセージから順次取り出すように構成したので、要求メッセージの欠損を防止できるとともに、高速性の要求される要求メッセージに対する処理から行うことができる効果がある。

【０１０６】請求項５の発明によれば、各スレッドが処理を実行する際に必要となるメモリ領域を各スレッド毎に予め確保しておき、各スレッドに処理を実行させる際に対応するメモリ領域を割り当てるように構成したので、ガベージコレクションを行う必要がなくなり、その結果、ガベージコレクションを原因とする要求メッセージに対する応答が妨げられるという事態が発生しなくなる効果がある。

【０１０７】請求項６の発明によれば、処理に必要となるスレッドを予め生成し、待機状態にしておくように構成したので、スタック領域の不足を理由に、要求メッセージに対する処理が遅延するのを防止できる効果がある。

【０１０８】請求項７の発明によれば、第１の受信部により受信されたサンプルデータをメモリに格納中に、処理実行部から所定のアドレスに格納されているサンプルデータの読み込み要求があった場合、その読み込み要求に係るアドレスが、現在サンプルデータを格納しているアドレスと一致するときのみその読み込み要求を却下するように構成したので、同じデータを同時に読み書きすることによって生じるデータ内容の矛盾を防止しつつデータの読み書きの並列処理が可能になり、その結果、高速に処理を行うことができる効果がある。

【０１０９】請求項８の発明によれば、所定の起動周期毎に周期起動型スレッドを起動させるとともに、その周期起動型スレッドが起動後所定時刻経過しても処理を終了しない場合には、その処理を中断させて獲得している資源を解放させるタイマスレッドを設けるように構成したので、他の周期起動型スレッドを所定の起動周期毎に確実に起動できるようになり、周期起動型スレッドのリアルタイム性を確保できる効果がある。

【０１１０】請求項９の発明によれば、タイマスレッドより優先度の低いスレッドがセマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にする優先度継承機能をそのセマフォに持たせるように構成したので、同じデータを同時に読み書きすることによって生じるデータ内容の矛盾を防止しつつ、タイマスレッドが速やかにセマフォを獲得できる結果、周期起動型スレッドのリアルタイム性を確保できるとともに、装置の信頼性が向上する効果がある。

【０１１１】請求項１０の発明によれば、共有資源を使用しているか否かを示すフラグ及びそのフラグを読み書きするためのセマフォを各スレッド毎に設けるとともに、タイマスレッドより優先度の低いスレッドが当該セマフォを獲得しているためにそのタイマスレッドが待ち状態になっている場合、一時的にその優先度の低いスレッドの優先度をそのタイマスレッドの優先度と同一にす

る優先度継承機能をそのセマフォに持たせるように構成したので、同じデータを同時に読み書きすることによって生じるデータ内容の矛盾を防止しつつ、タイマスレッドが速やかにセマフォを獲得できる結果、周期起動型スレッドのリアルタイム性を確保できるとともに、装置の信頼性が向上する効果がある。

【図面の簡単な説明】

【図１】この発明の実施例１によるマルチスレッド・サーバを示す構成図である。

【図２】実施例２における送信部の詳細を示す構成図である。

【図３】実施例３における送信部の詳細を示す構成図である。

【図４】実施例４における第２の受信部の詳細を示す構成図である。

【図５】スループット機構の構成を示す構成図である。

【図６】スレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。

【図７】スレッドプール３１内のチャイルドスレッドの動作を示すフローチャートである。

【図８】スレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。

【図９】スレッドプール３１内のチャイルドスレッドの動作を示すフローチャートである。

【図１０】スレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。

【図１１】スレッドプール３１内のチャイルドスレッドの動作を示すフローチャートである。

【図１２】スレッドプール３１内のチャイルドスレッドが有する起動情報の構成図である。

【図１３】スレッドプール３１内のチャイルドスレッドの動作を示すフローチャートである。

【図１４】リングバッファ１３及びディスク１５に対して複数のスレッドが書き込み及び読み出しを行う動作を説明する構成図である。

【図１５】この発明の実施例７によるマルチスレッド・サーバを示す構成図である。

【図１６】スレッド管理テーブルの内容を説明する説明図である。

【図１７】タイマスレッドの動作を示すフローチャートである。

【図１８】マザースレッドがスレッド管理テーブルに周期起動型スレッドを登録する動作を示すフローチャートである。

【図１９】マザースレッドがスレッド管理テーブルから周期起動型スレッドの登録を削除する動作を示すフローチャートである。

【図２０】各周期起動型スレッドの動作を示すフローチャートである。

【図２１】スレッド管理テーブルの内容を説明する説明図である。

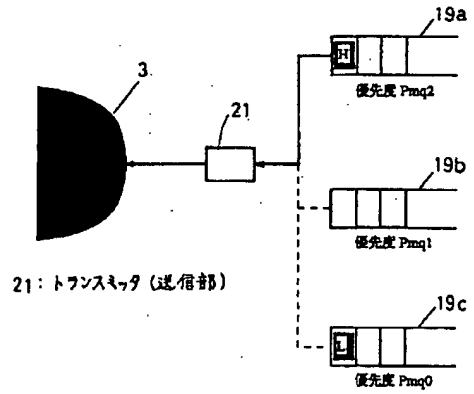
【図２２】従来のマルチスレッド・サーバを示す構成図である。

【図２３】従来の送信部の詳細を示す構成図である。

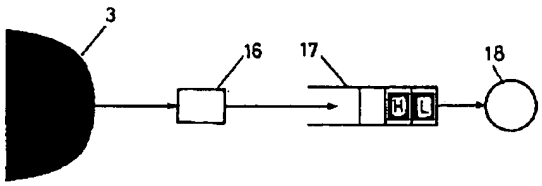
【符号の説明】

１　プラント
２　クライアント（外部装置）
３　ネットワーク（外部装置）
６、７、８　チャイルドスレッド（処理実行部）
１１　ＦＩＦＯキュー（第１の受信部）
１３　リングバッファ（メモリ）
１５　ディスク（メモリ）
１６　レシーバ（第２の受信部）
１７　ＦＩＦＯキュー（第２の受信部）
１８　マザースレッド（処理実行部）
１９　ＦＩＦＯキュー（送信部）
２０、２１　トランスミッタ（送信部）
２１ａ、２１ｂ、２１ｃ　送信スレッド（送信部）
３２、３３、３４　メモリ領域
６１　タイマスレッド
６２　スレッド管理テーブル（資源）
６３～６５　セマフォ

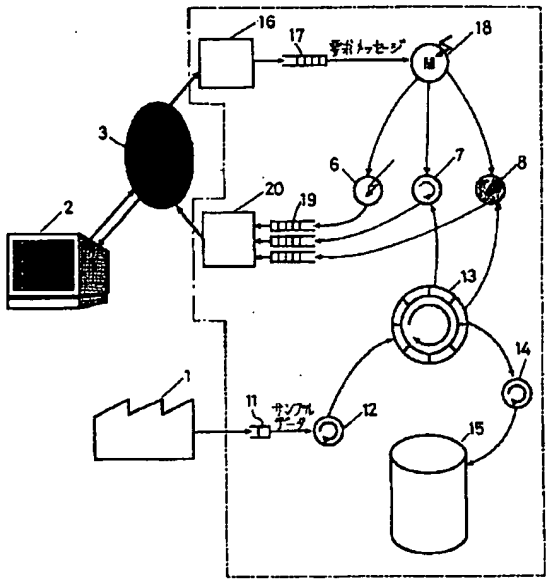
【図２】



【図４】

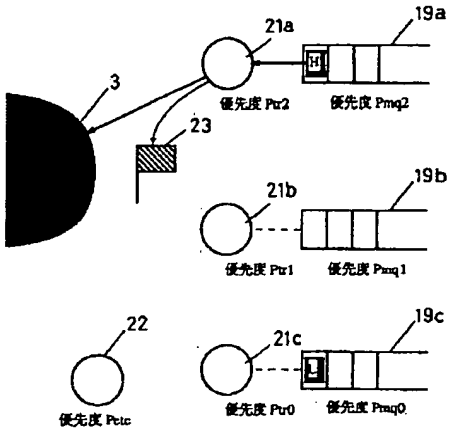


【図１】



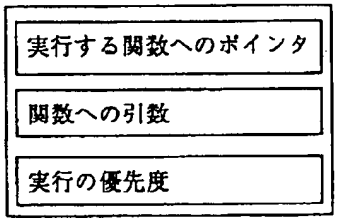
1：クライアント
2：クライアント（外部装置）
3：ネットワーク（外部装置）
6,7,8：チャイルドスレッド（処理実行部）
11：FIFOキュー（第1の受信部）
13：リングバッファ（メモリ）
15：ディスク（メモリ）
16：レシーバ（第2の受信部）
17：FIFOキュー（第2の受信部）
18：マザースレッド（処理実行部）
19：FIFOキュー（送信部）
20：トランスミッタ（送信部）

【図３】



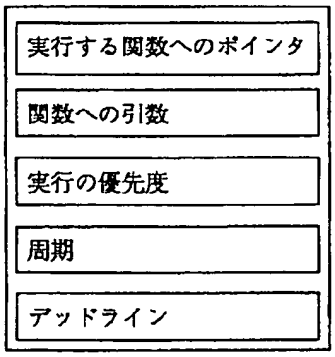
21a, 21b, 21c：送信スレッド（送信部）

【図６】



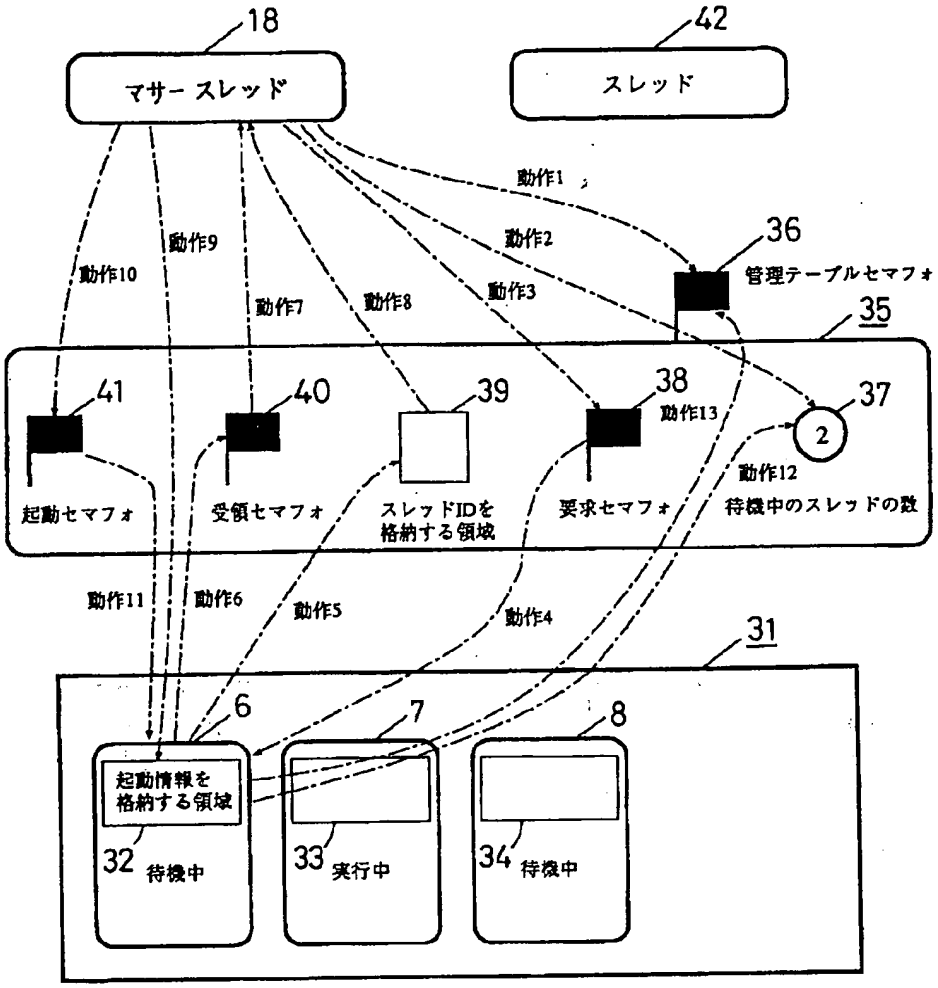
リアルタイム性を要しないスレッドの起動情報

【図８】



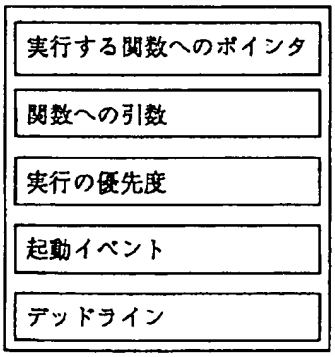
周期起動型スレッドの起動情報

【図５】



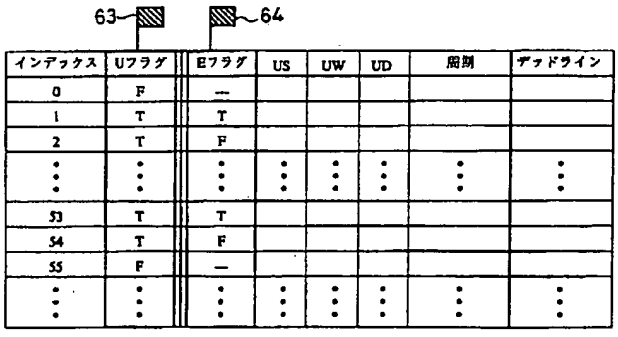
32,33,34：メモリ領域

【図１０】



イベント起動型スレッドの起動情報

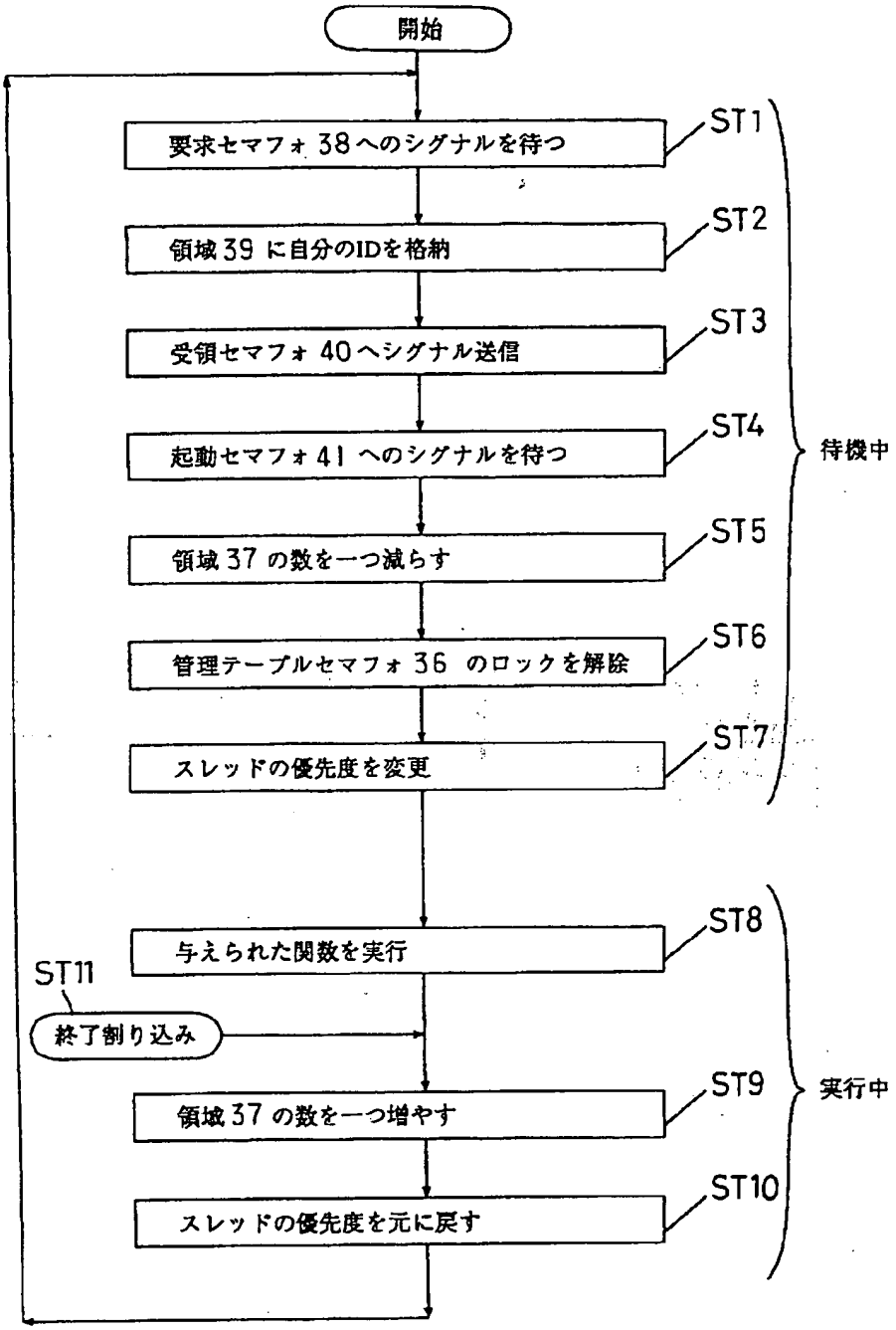
【図１６】



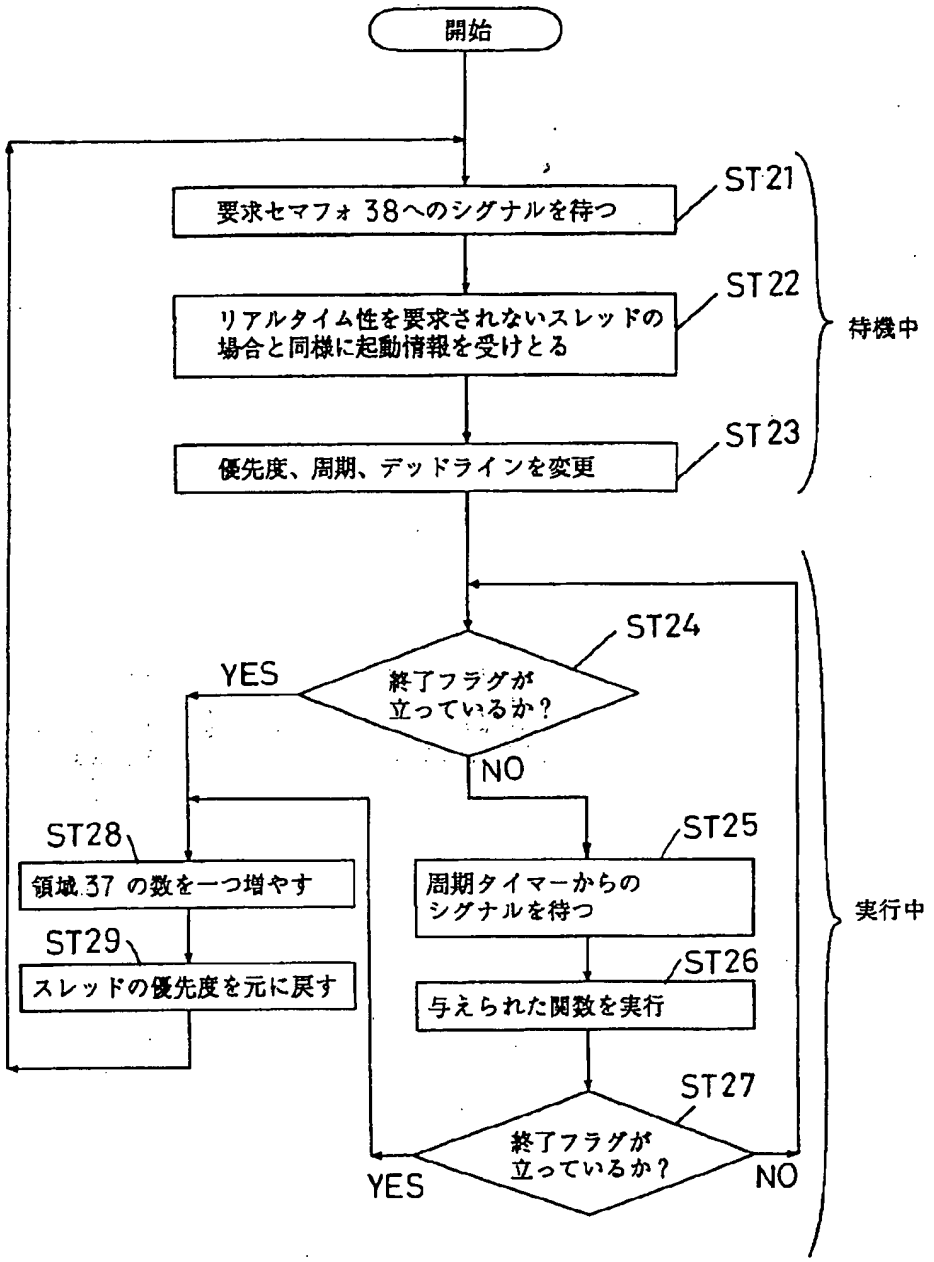
T ：TRUE
F ：FALSE
—：任意

63，64：セマフォ

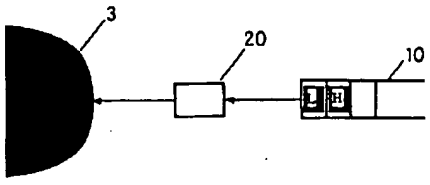
【図７】


開始
要求セマフォ 38 へのシグナルを待つ
ST1
領域 39 に自分のIDを格納
ST2
受領セマフォ 40 へシグナル送信
ST3
起動セマフォ 41 へのシグナルを待つ
ST4
領域 37 の数を一つ減らす
ST5
管理テーブルセマフォ 36 のロックを解除
ST6
スレッドの優先度を変更
ST7
待機中
与えられた関数を実行
ST8
ST11
終了割り込み
領域 37 の数を一つ増やす
ST9
スレッドの優先度を元に戻す
ST10
実行中

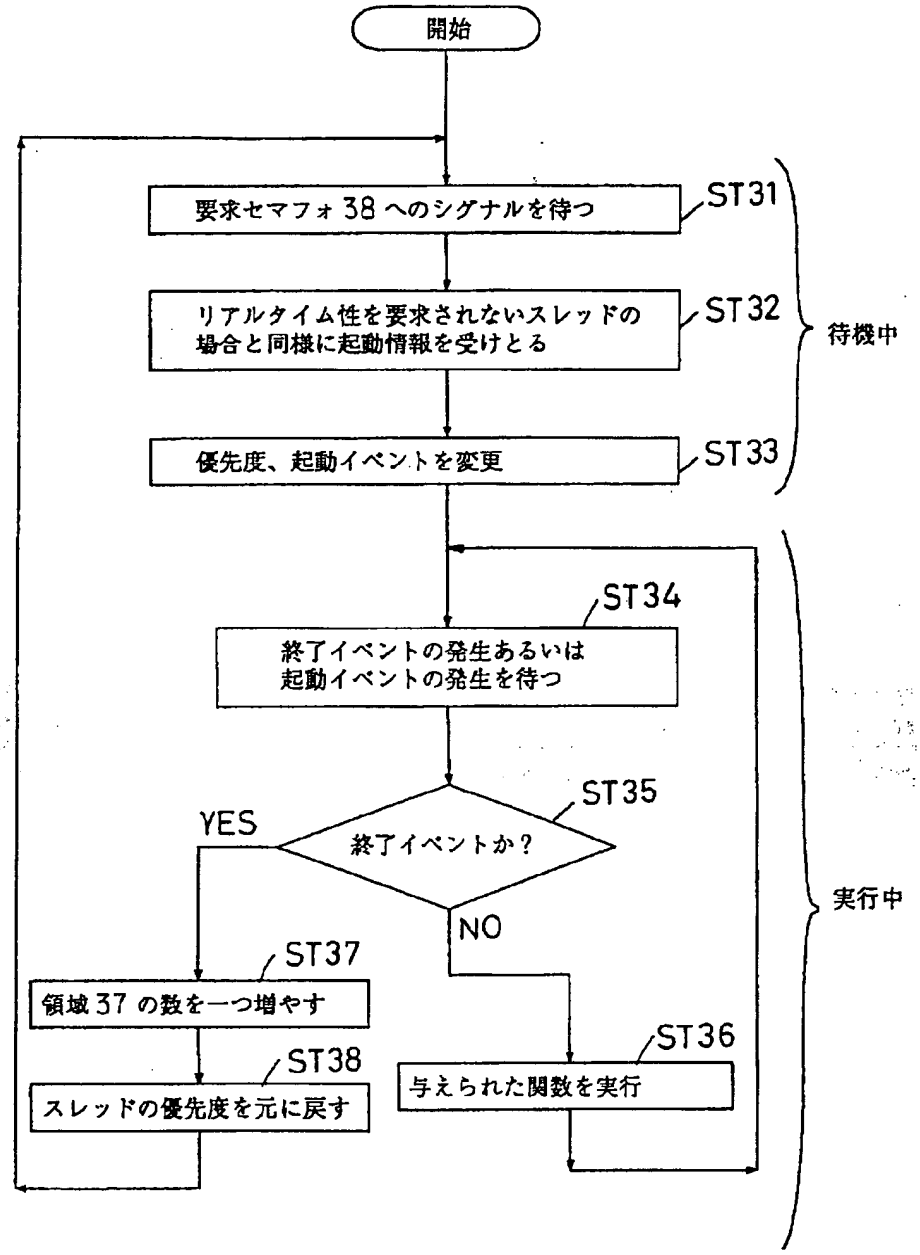
【図９】


開始
要求セマフォ 38へのシグナルを待つ
ST21
リアルタイム性を要求されないスレッドの場合と同様に起動情報を受けとる
ST22
優先度、周期、デッドラインを変更
ST23
待機中
ST24
終了フラグが立っているか？
YES
NO
ST28
領域37 の数を一つ増やす
ST29
スレッドの優先度を元に戻す
ST25
周期タイマーからのシグナルを待つ
ST26
与えられた関数を実行
ST27
終了フラグが立っているか？
YES
NO
実行中

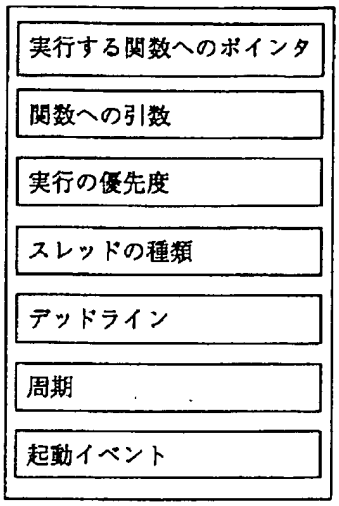
【図２３】


3
20
10

【図１１】

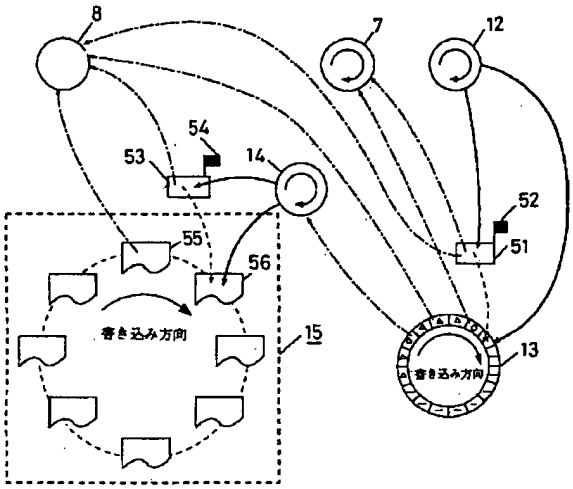

開始
要求セマフォ 38 へのシグナルを待つ
ST31
リアルタイム性を要求されないスレッドの
場合と同様に起動情報を受けとる
ST32
優先度、起動イベントを変更
ST33
待機中
ST34
終了イベントの発生あるいは
起動イベントの発生を待つ
ST35
終了イベントか？
YES
NO
ST37
領域 37 の数を一つ増やす
ST38
スレッドの優先度を元に戻す
ST36
与えられた関数を実行
実行中

【図１２】



スレッドプール内のスレッドの起動情報

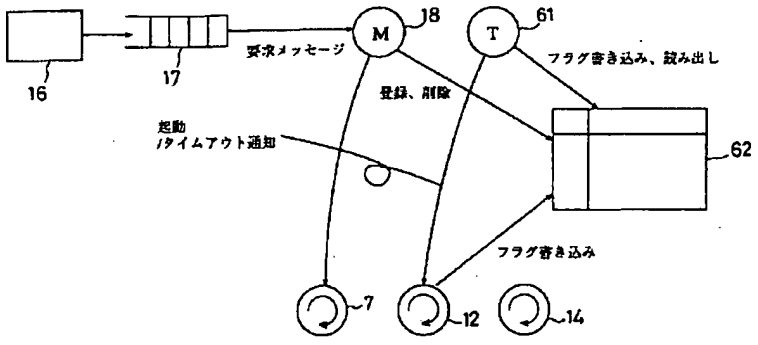
【図１４】



× 書き込み中のインデックス
○ 最新のデータを格納しているインデックス
△ ディスク上に読み出し可能なバックアップのないインデックス
－ ディスク上に読み出し可能なバックアップのあるインデックス

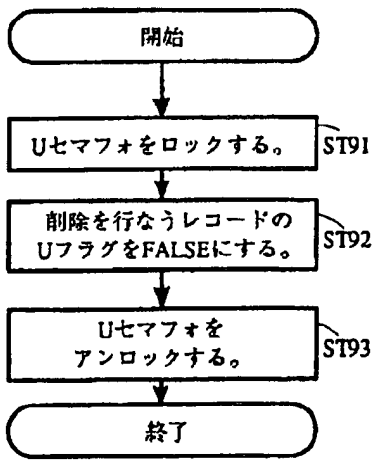
リングバッファ、ディスク等への書き込みを表す。
リングバッファ、ディスク等からの読み出しを表す。
リングバッファのインデックスまたはファイルをさすことを表す。

【図１５】

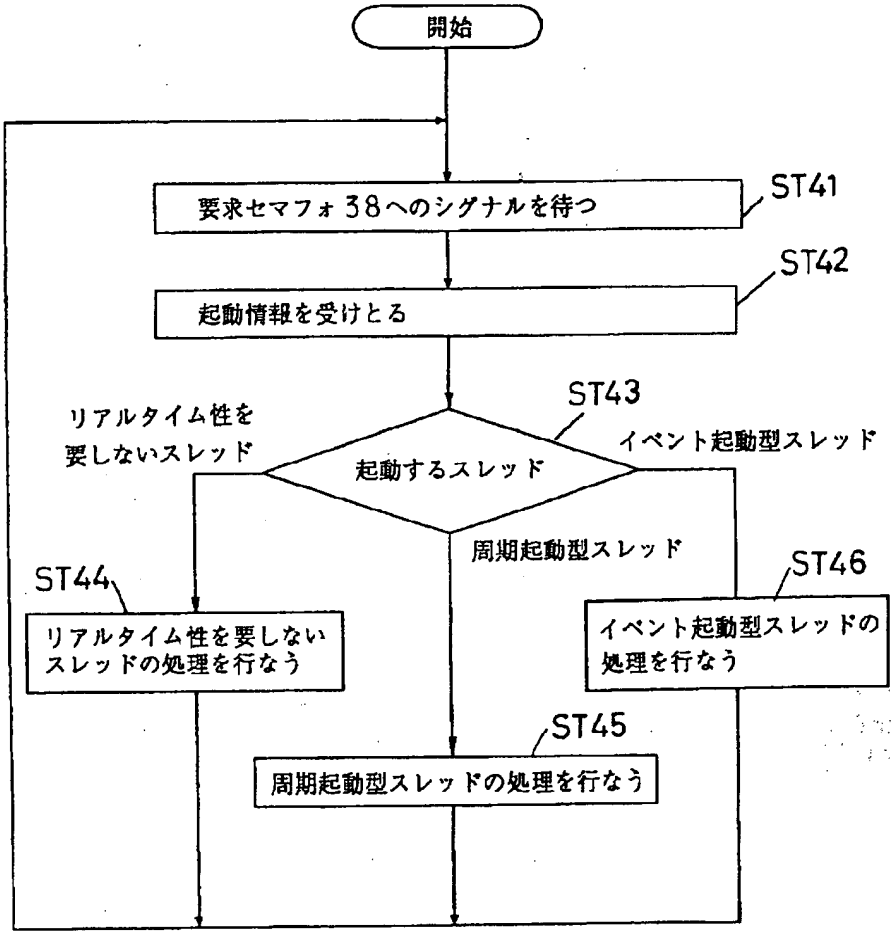


61：タイマスレッド
62：スレッド管理テーブル（資源）

【図１９】



【図１３】



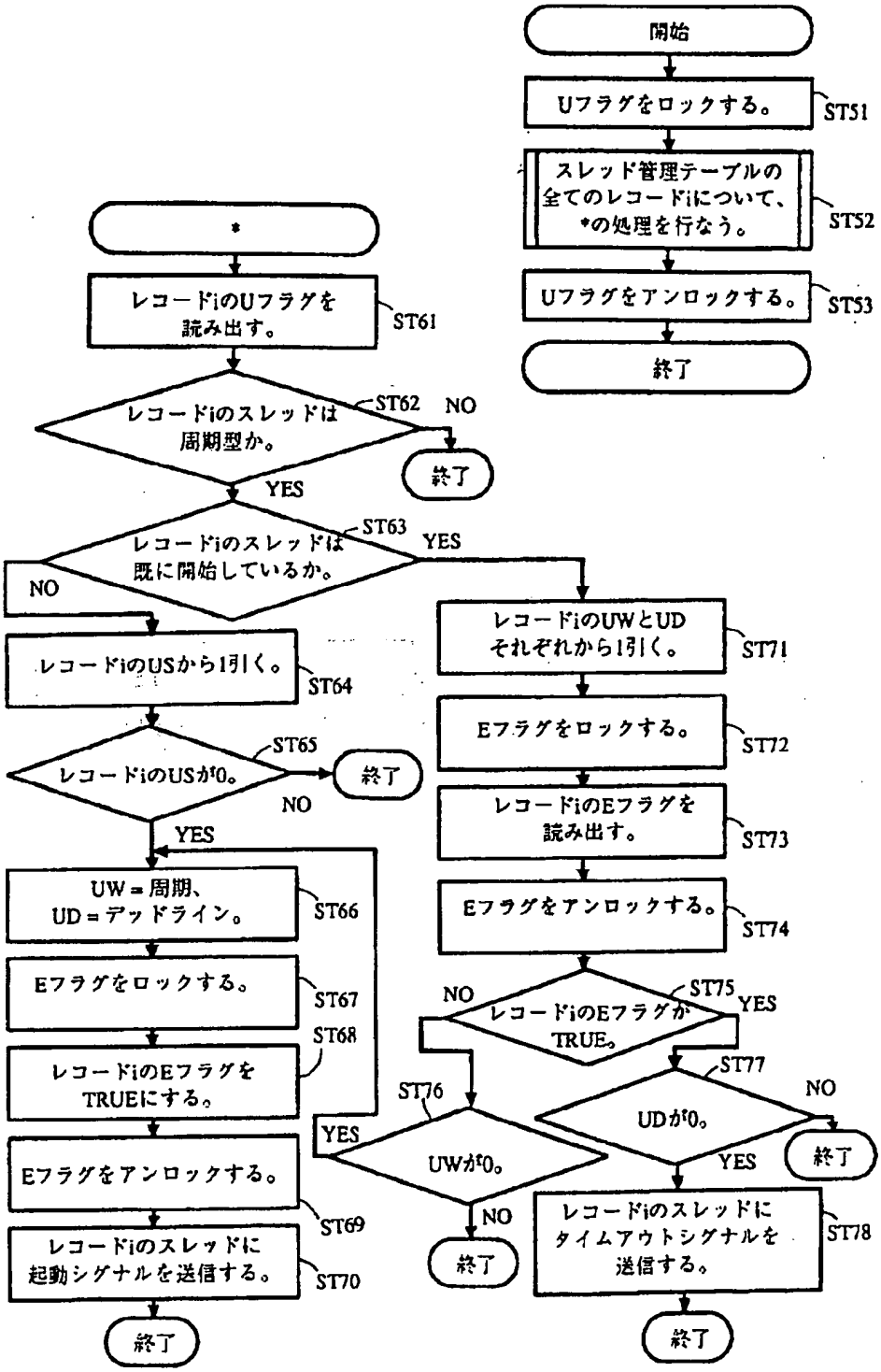
【図２１】

インデックス	Uフラグ	Eフラグ	US	UW	UD	周期	デッドライン	Sフラグ	
0	F	—						F	～65
1	T	T						F	～65
2	T	F						T	～65
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
53	T	T						T	～65
54	T	F						F	～65
55	F	—						F	～65
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

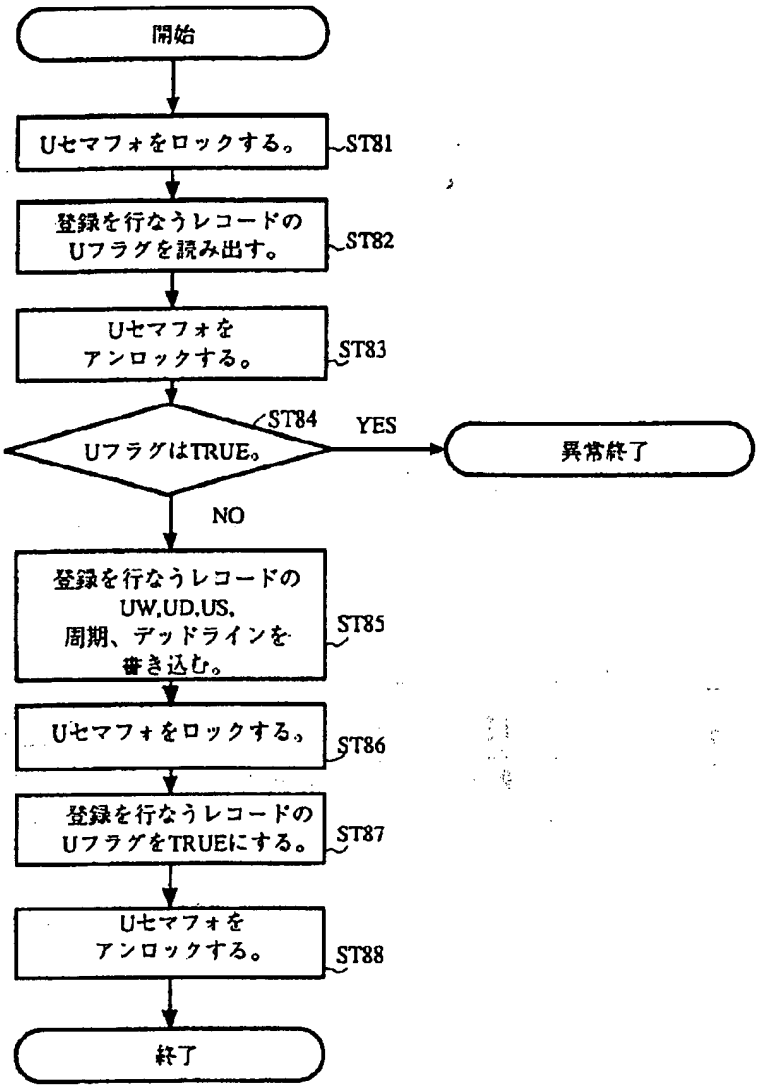
T　:TRUE
F　:FALSE
—:任意

65:セマフォ

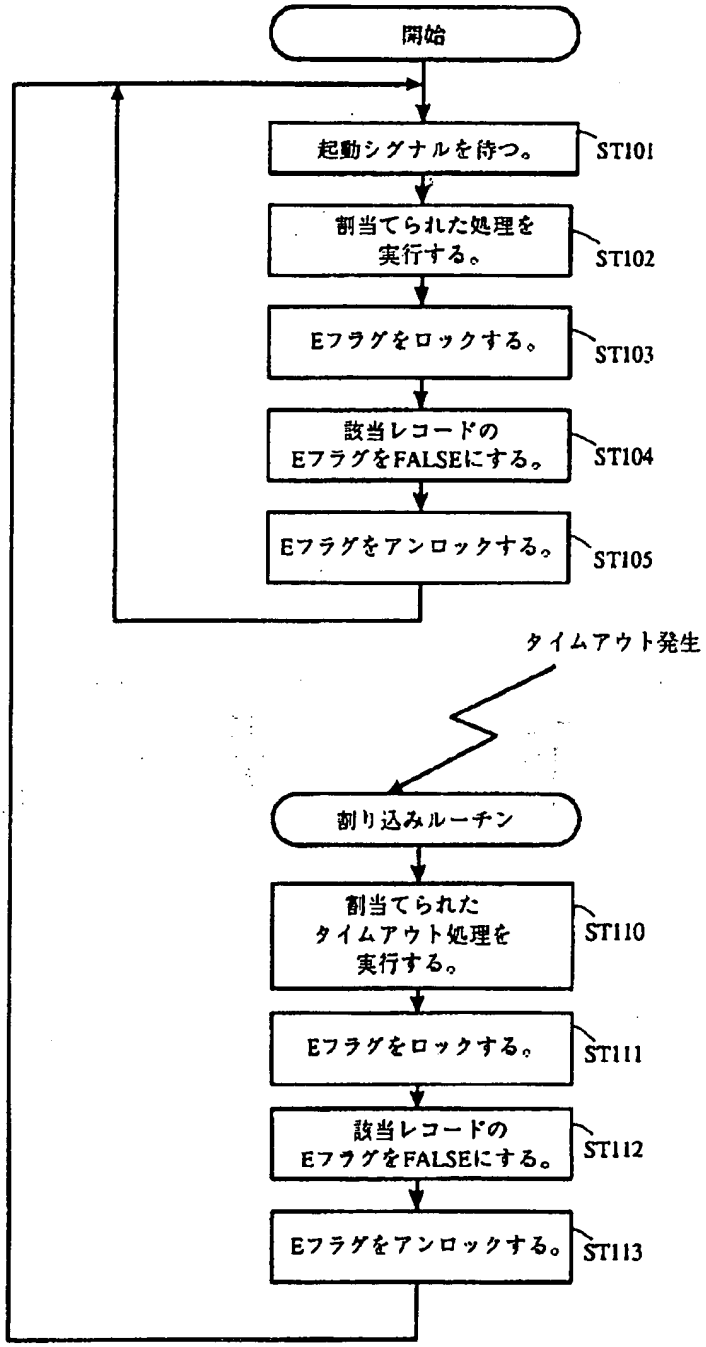
【図１７】


開始
Uフラグをロックする。
ST51
スレッド管理テーブルの全てのレコードiについて、*の処理を行なう。
ST52
Uフラグをアンロックする。
ST53
終了
*
レコードiのUフラグを読み出す。
ST61
レコードiのスレッドは周期型か。
ST62
NO
終了
YES
レコードiのスレッドは既に開始しているか。
ST63
YES
NO
レコードiのUSから1引く。
ST64
レコードiのUSが0。
ST65
終了
NO
YES
UW＝周期、UD＝デッドライン。
ST66
Eフラグをロックする。
ST67
レコードiのEフラグをTRUEにする。
ST68
Eフラグをアンロックする。
ST69
レコードiのスレッドに起動シグナルを送信する。
ST70
終了
レコードiのUWとUDそれぞれから1引く。
ST71
Eフラグをロックする。
ST72
レコードiのEフラグを読み出す。
ST73
Eフラグをアンロックする。
ST74
NO
レコードiのEフラグがTRUE。
ST75
YES
ST76
UWが0。
YES
NO
終了
ST77
UDが0。
NO
YES
終了
レコードiのスレッドにタイムアウトシグナルを送信する。
ST78
終了

【図１８】


開始
Uセマフォをロックする。
ST81
登録を行なうレコードの
Uフラグを読み出す。
ST82
Uセマフォを
アンロックする。
ST83
ST84
UフラグはTRUE。
YES
異常終了
NO
登録を行なうレコードの
UW,UD,US,
周期、デッドラインを
書き込む。
ST85
Uセマフォをロックする。
ST86
登録を行なうレコードの
UフラグをTRUEにする。
ST87
Uセマフォを
アンロックする。
ST88
終了

【図２０】


開始
起動シグナルを待つ。
ST101
割当てられた処理を
実行する。
ST102
Eフラグをロックする。
ST103
該当レコードの
EフラグをFALSEにする。
ST104
Eフラグをアンロックする。
ST105
タイムアウト発生
割り込みルーチン
割当てられた
タイムアウト処理を
実行する。
ST110
Eフラグをロックする。
ST111
該当レコードの
EフラグをFALSEにする。
ST112
Eフラグをアンロックする。
ST113

【図２２】